2009 IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)

IEEE Catalog Number CFP09010-CDR
ISBN 978-1-4244-4928-6
ISSN 1526-7539

Additional copies of this publication are available from
IEEE Operations Center
445 Hoes Lane
Piscataway, NJ 08854-4150 USA
+1 800 678 IEEE (+1 800 678 4333)
+1 732 981 1393
+1 732 981 9667 (FAX)
email: `customer.services@ieee.org`

Jeremy T. Bradley (corresponding editor)
James M. Conrad
A.J. Field
Uli Harder
William J. Knottenbelt
George F. Riley

Department of Computing
Imperial College London
Huxley Building
180 Queen's Gate
London
SW7 2RH

Adaptive Admission Control Algorithm in a QoS-aware Web System

Katja Gilly*, Carlos Juiz†, Salvador Alcaraz* and Ramon Puigjaner†
*Miguel Hernández University,
Departamento de Física y Arquitectura de Computadores,
Avda. de la Universidad, 03202 Elche (Spain).
†University of Balearic Islands,
Departament de Ciències Matemàtiques i Informàtica,
Carretera de Valldemossa, km 7.5, 07071 Palma de Mallorca (Spain).

***Abstract*—We address two issues in this work: on one hand, we consider and compare five throughput predictors to be used in a Web system in order to estimate its future performance and, on the other hand, we propose an adaptive QoS-aware admission control algorithm that is based on a resource allocation scheme that includes a throughput predictor. In order to obtain a low overhead, the monitoring of the arrival traffic to the Web system is done following an adaptive time slot scheduling based on the burstiness factor that we defined in a previous work. Simulation results of the behaviour of the five throughput predictors and the admission control algorithm are presented and discussed.**

I. Introduction

It has been widely studied that Internet traffic is self similar and that sudden bursts of packets can reach a point in a network infrastructure that offers Web services. This affects considerably to the performance of the system in case it is not prepared to process that increase in the demand.

We propose an adaptive admission control algorithm that prevents the system from a not expected overload by predicting the throughput of the Web servers. We have considered five throughput predictors in this work in order to compare them once included in our admission control algorithm. The problem of allocating the resources of a Web System that considers QoS is also addressed. Our algorithm is prepared to be implemented in a cluster of Web servers to increase the scalability of the solution.

II. Aiming for low overhead

In order to obtain a low overhead, we plan the invocation times of the algorithm based on the arrival rate. We defined a burstiness factor, $b(k)$, in a previous work [1]. This factor varies its value in a range of [0,1] and gives an indication of the burstiness perceived at the entry point of the system. The entry point can be defined as the front-end of the system, or each of the Web servers that are currently attending requests. Based on this factor, we also defined an adaptive time slot scheduling in [1], that sets the times the admission control algorithm is invoked. Hence, we divide the time in slots (k) of different durations ($d(k)$) during the experiment. The burstiness detected in the system influences in the execution of the admission control algorithm in this way: when an increase in the burstiness is detected, then the algorithm is invoked more frequently, and viceversa.

III. System Architecture

The system architecture proposed is based on Web cluster-based network servers and includes a front-end Web switch. A Web switch is normally described as a content-aware switch that can dencapsulate the requests up to the application level and classify them on the basis of this information. The cluster of Web servers is locally connected to the Web switch in a two-tier organisation. Each Web server attends the requests that ask for static files, namely static requests and the Application/Database server is accessed when the request asks for a Web page that needs to retrieve dynamic content (dynamic requests).

IV. QoS-awareness

Different classes of requests are distinguished in the algorithm. We define the priority of each class of requests by setting a fraction of the utilisation of the whole Web system for each class. Hence, we consider a set of classes, $C = \{c_1, c_2, \dots, c_r\}$, and define for them a normalised utilisation value in a decreasing order. Hence, the class of requests that represent c_1 have more priority than the class c_2, and so on. The sum of the utilisation values of all the classes is equal to 1, that represents the whole utilisation of the Web system. The admission control algorithm defines how the different classes of requests are distributed among the Web servers.

V. Throughput Prediction

The admission control algorithm is based on throughput prediction for a Web system. We have defined five throughput predictors that give us the trend of the system behaviour and permit the admission control algorithm to take decisions that maintain the performance of the system. A previous version of the predictors P1-P3 were introduced in [2]. We have extended the research in throughput prediction in order to obtain better results. Let us briefly present these five predictors.

A. P1: based on filtering

This predictor is basically a moving average between the estimated value of the throughput in the last slot and the harmonic mean of the throughput obtained in the last two slots.

$$\hat{x}_1(k+1) = (1 - a(k+1)) \cdot \hat{x}_1(k) + a(k+1) \cdot \frac{2}{\frac{1}{x(k)} + \frac{1}{x(k-1)}}$$

The value $a(k+1)$ is obtained by the total observation time, T, and the duration of the next slot, $d(k+1)$:

$$a(k+1) = \frac{2 \cdot T - d(k+1)}{2 \cdot T + d(k+1)}$$

B. P2: based on burstiness

This predictor includes the burstiness factor we described in Section II in the prediction of the throughput.

$$\hat{x}_2(k+1) = \hat{x}_2(k) - (\beta(k+1) \cdot u(k))$$
$$\beta(k+1) = (b(k) - b(k-1)) \cdot |x(k) - x(k-1)|$$

The goal of this estimator is to decrease the prediction when a burst of requests is detected, depending on the current utilisation of the servers.

C. P3: based on filtering and burstiness

The harmonic mean of the predictor 1 and 2 is considered as the third predictor.

D. P4: based on least mean square (LMS)

We have also considered the Least Mean Square (LMS) algorithm [3] to predict the throughput. LMS introduces an iterative procedure that makes successive corrections to a weight vector that minimises the mean square error.

Let $\underline{w}(k+1)$ denote the weight vector of the LMS filter that is computed at the k slot. The operation can be resumed by the following recursive operation:

$$\underline{w}(k+1) = \underline{w}(k) + \mu \cdot [x(k) - \hat{x}_5(k)] \cdot \underline{x}(k),$$

where M is the number of tap weights used in the adaptive transversal filter, μ is the step-size parameter and the vectors $\underline{w}(k)$ and $\underline{x}(k)$ are defined as:

$$\underline{w}(k) = [w_0(k), w_1(k), ..., w_{M-1}(k)]^T$$
$$\underline{x}(k) = [x(k), x(k-1), ..., x(k-M+1)]^T$$

The predicted value of throughput is obtained by linear prediction.

E. P5: based on normalised least mean square (NLMS)

In order to avoid the sensibility of the LMS algorithm to the scaling of its input $x(k)$, the NLMS method normalises the previous expression by dividing the vector $\underline{x}(k)$ by the square of its Euclidean norm.

$$\underline{\hat{w}}(k+1) = \underline{\hat{w}}(k) + \mu \cdot [x(k) - \hat{x}_6(k)] \cdot \frac{\underline{x}(k)}{\|\underline{x}(k)\|^2}$$

VI. Resource Allocation

We base the resource allocation strategy on the throughput obtained by the estimator, and the service times ($\delta(k)$) monitored in the Web and Application/Database servers. Hence, we can predict the utilisation that each class of traffic will have in each server of the Web system with this expression:

$$\hat{u}(k+1) = \hat{x}(k+1) \cdot \delta(k)$$

This prediction does not include the SLA we have defined for each class of request. Therefore, we normalise the predicted utilisation to guarantee the priority requirements of each class. For a better understanding of this modification, we include two subscripts to the following expressions: i and j. The first one, i, represents a Web server and Application/Database server set, and the second one, j, represents a class of traffic. Depending on the content that the request asks for, it is mainly attended by the Web server (static request) or by the Application/Database server (dynamic request).

The normalisation of the utilisation estimation is done to distribute the 100% of the available capacity of the Web system between all the classes of requests, being N the number of Web and Application/Database server sets that are included in the Web system:

$$\hat{u}'_{i,j}(k+1) = (c_j \cdot N) \frac{\hat{u}_{i,j}(k)}{\sum_{\forall i} \hat{u}_{i,j}(k)}$$

With this modification, we assure that each traffic class has reserved the utilisation of the Web system that corresponds to its SLA. To obtain the number of requests that are going to be accepted for each class in each server during the following slot, we make the inverse operation and multiply the obtained throughput by the duration of the next slot in order to obtain the number of requests:

$$\gamma_{i,j}(k+1) = \frac{\hat{u}'_{i,j}(k+1)}{\delta(k)} \cdot d(k+1)$$

During the next slot, each Web and Application/Database server counts the number of accepted requests. When $\gamma_{i,j}(k+1)$ is reached, the server stops attending that class of requests until the next invocation of the admission control algorithm.

VII. Results

The five throughput predictors have been tested in the resource allocation algorithm. We have configured a simulation scenario in OPNET Modeler that consists of 5 Web and Application/Database servers that attend two different classes of requests. The SLA specified for each class is $c_1 = 0.625$ for class-1 requests and $c_2 = 0.375$ for class-2. We have stressed the Web system with requests coming from by 30, 40, 50, 60, 70, 80, 90 and 100 Web clients. Obviously, as the load increases in the Web system, the algorithm needs to reject more requests.

We have considered a Pareto user think time in the Web clients, and the session duration and the session inter-repetition time are modelled according to a exponential distribution. The file size has been obtained from the logs of the 24th of June of the 1998 World Cup Web Site.

Each simulation has been run for 2000 seconds with 4 different seeds. As we have introduced the same proportion of static and dynamic requests in the system, the bottleneck of the architecture is the Application/Database server. In fact, the admission control algorithm limits the CPU utilisation of the Application/Database servers and then, starts to reject class-2 dynamic requests with 40 clients, as it can be observed in the lower part of Figure 1a). In the upper part of this figure, we see that Class-1 requests are also rejected when there are 70 clients asking for dynamic content.

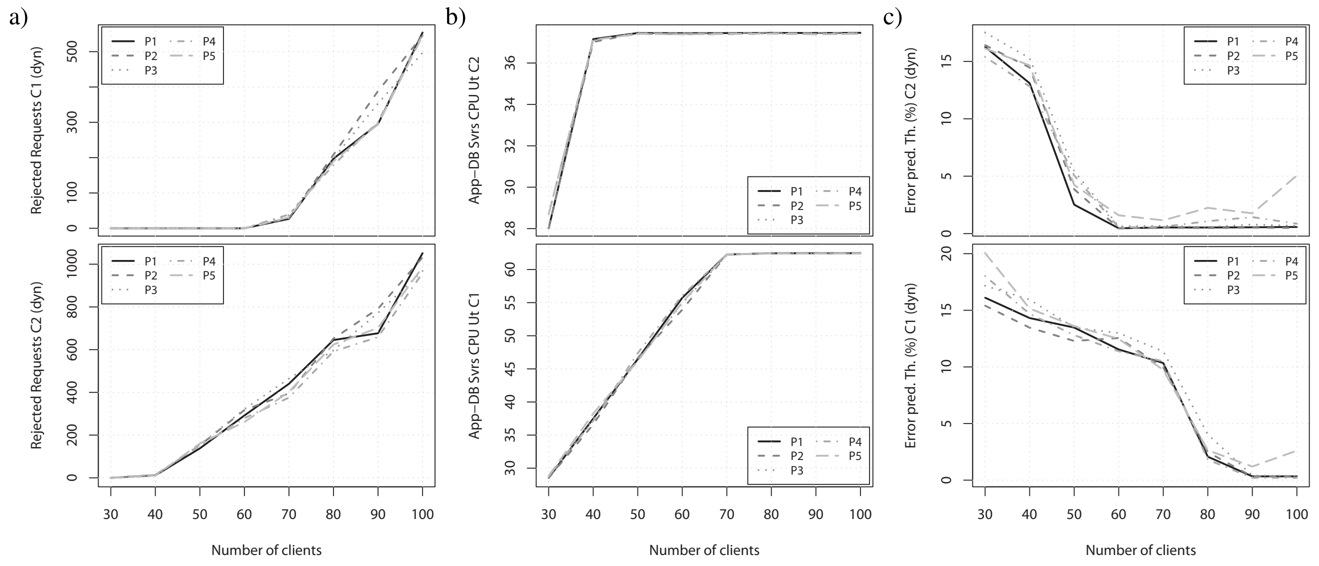

Fig. 1. a) Rejection of dynamic requests; b) Back-end server Utilisation for requests that ask for dynamic content and c) Error of Throughput Predictions of dynamic requests.

The 90th percentile of the CPU utilisation mean of all the Application/Database servers for each traffic class is represented in Figure 1b). The maximum utilisation for class-2 requests is almost reached with 40 clients, while for class-1 requests the maximum is obtained with 70 clients. Hence, Application/Database servers are at 100% of their utilisation with 70 clients.

In order to compare the goodness of the predictions made by the five predictors we defined in Section V, we have also computed the error obtained once the predictor has been included in the admission control algorithm. In Figure 1c), we can observe that the errors of the predictors decrease when the servers are reaching their maximum utilisation for each traffic class. This is due to the fact that most of the requests of that traffic class are rejected, hence, the throughput prediction is near to zero.

We have also considered the response time of the requests that stress the bottleneck of the system, that are the dynamic requests. It can be observed in Figure 2 the response time of dynamic requests of class-1. The differences among the curves begin at 60 clients, when the Application/Database server starts to be overloaded. The predictor P4 stands out for its lowest values of the response time.

VIII. Conclusions and Open Problems

We introduce a low overhead admission control algorithm that bases its decisions on the values obtained by a throughput predictor. The invocation times of the algorithm are adaptively scheduled depending on the burstiness detected in the system in order to reduce the overhead of the algorithm. Five throughput predictors are introduced in this work and their behaviour in the admission control algorithm is compared under a simulation scenario in OPNET Modeler. The resource allocation algorithm adaptively distributes the utilisation of the servers among the different classes of requests. The results show that the algorithm guarantees the service specified in the SLA with all the throughput predictors, but show differences in the response times of the requests that ask for dynamic content as the bottleneck of the system starts to be overloaded.

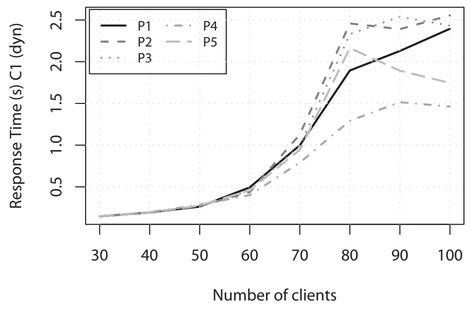

Fig. 2. Response Time

Simulation experiments that include a more variable workload are planned as a future work in order to verify the results of this work and confirm that the predictor based in LMS is the one that best suits in the algorithm.

IX. Acknowledgments

This work has been partially funded by the Spanish Ministry of Education and science under Grant TIN2006-02265.

References

[1] K. Gilly, S. Alcaraz, C. Juiz, and R. Puigjaner, “Analysis of burstiness monitoring and detection in an adaptive web system,” *Computer Networks*, vol. 53, pp. 668–679, 2009.
[2] K. Gilly, S. Alcaraz, C. Juiz, and R. Puigjaner, “Comparison of predictive techniques in cluster-based network servers with resource allocation,” in *proc. of the IEEE MASCOTS*, 2004.
[3] S. Haykin, *Adaptive Filter Theory*. Prentice-Hall, 1991.